Universidad Nacional de La Matanza

Programación Web 3

UNLaM - Tecnicatura en Desarrollo Web

Trabajo Práctico de Investigación

Título: *SignalR*

Índice

Integrantes

Aibar, Ezequiel
Matto, Rodrigo
Reta, Lucas
Ruiz, Ariel
Siciliano, Aldana
Tonietti, Lucia

Objetivo

El objetivo del presente documento, es trabajar con SignalR, especificando sus principales funciones y métodos de implementación

Situación Actual

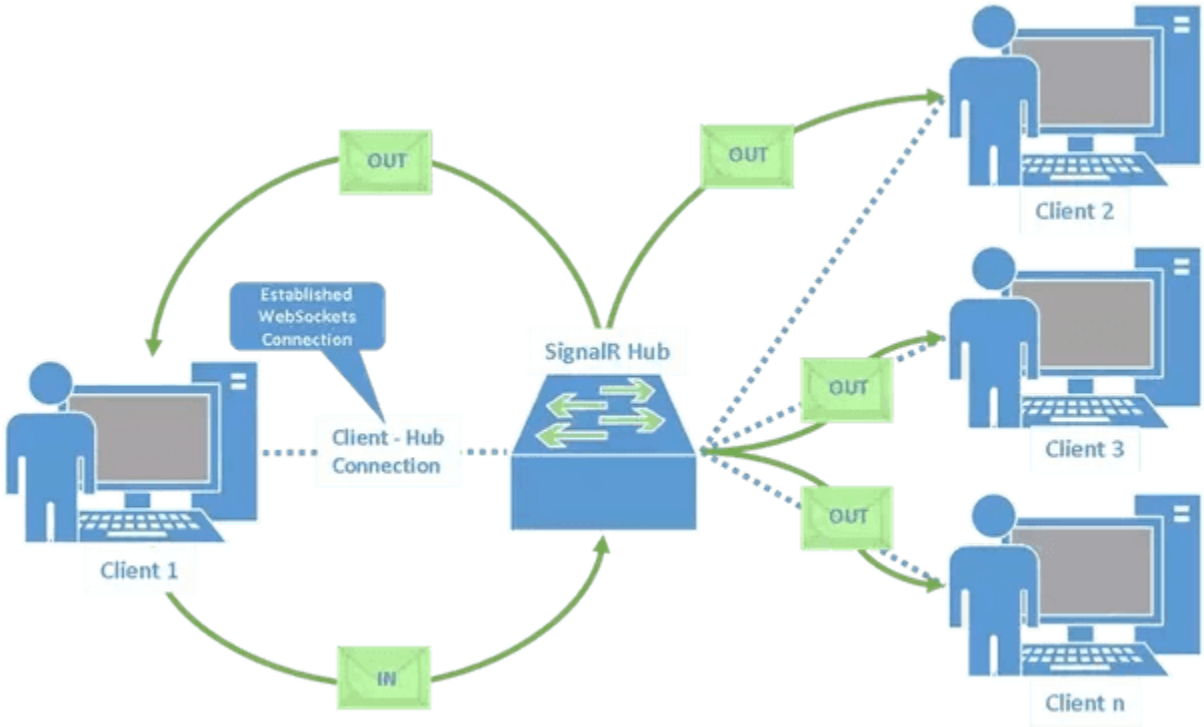
La arquitectura de software, que predominan en aplicaciones web, son librerías que potencien el uso de las páginas y apps, de ahí predomino el pensar una arquitectura ambiciosa en cuanto a rendimiento y escalabilidad. En el cual el rendimiento del servidor y cliente tengan una comunicación en línea, para eso Microsoft, desarrollo SignalR

SignalR

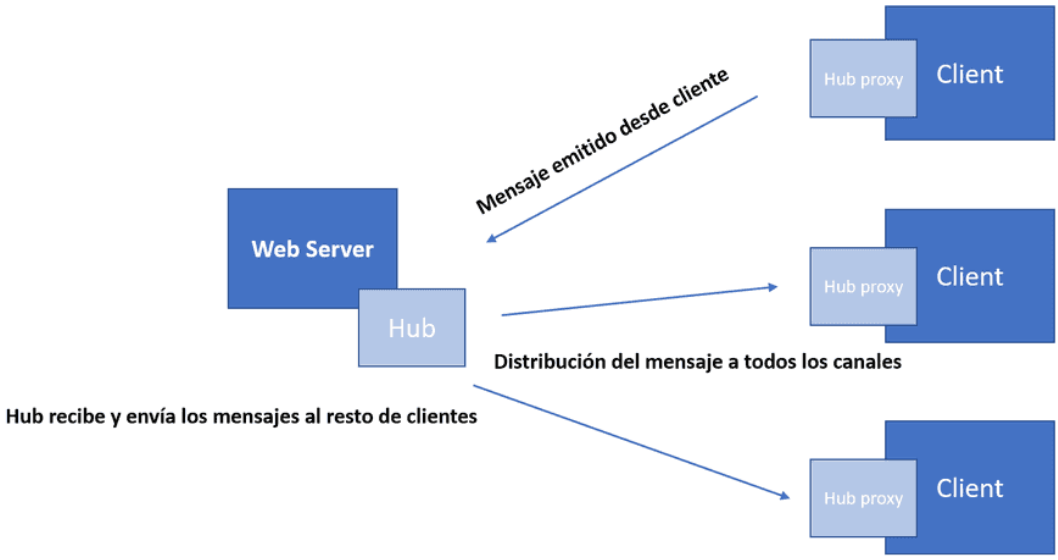
SignalR es una biblioteca para ASP.NET desarrolladores que simplifican el proceso de agregar funcionalidad web en tiempo real a las aplicaciones.

La funcionalidad web en tiempo real es la capacidad de que el código del servidor inserte contenido en los clientes conectados al instante a medida que esté disponible, en lugar de tener que esperar al servidor a que un cliente solicite nuevos datos.

SignalR proporciona una API sencilla para crear llamadas a procedimientos remotos de servidor a cliente (RPC) que llaman a funciones de JavaScript en exploradores cliente (y otras plataformas cliente) desde código .NET del lado servidor. SignalR también incluye la API para la administración de conexiones (por ejemplo, eventos de conexión y desconexión) y las conexiones de agrupación.



Como vemos en la figura, cada mensaje que se envía al Hub de SignalR, este lo distribuye por el canal indicado en la petición, a cada uno de los clientes conectados (Hub Proxy).



Además, de esto puede controlar automáticamente la administración de conexiones y permitir difundir mensajes a todos los clientes conectados de forma simultánea, como un salón de chat. También se pueden enviar mensajes a clientes concretos. La conexión entre el cliente y el servidor es persistente, a diferencia de una conexión HTTP clásica, que se vuelve a establecer para cada comunicación. Dicha conexión, es persistente a diferencia de una conexión HTTP clásica que se vuelve a establecer para cada petición.

Para lograr el envío de actualizaciones en tiempo real desde el servidor al cliente SignalR utiliza varias técnicas o mecanismos de comunicación. Esto se denominan transportes:

WEBSOCKETS

Eventos enviados por el servidor

Sondeos Largos

Long Polling

Transportes

SignalR utiliza el nuevo transporte WebSocket cuando está disponible y recurre a transportes anteriores cuando es necesario, a continuación describimos cada uno.

WebSockets: Tecnología de comunicación bidireccional que permite a los servidores enviar información a los clientes de forma asíncrona, es decir enviar información sin tener que esperar una respuesta inmediata. Siempre que sea posible SignalR optara por esta opción, ya que ofrece una latencia muy baja y una alta eficiencia. Tiene los requisitos más estrictos; sólo se admite en las versiones más recientes de Microsoft Internet Explorer, Google Chrome y Mozilla Firefox, y solo tiene una implementación parcial en otros exploradores, como Opera y Safari.

Server-Sent Events (Eventos enviados por el servidor): Tecnología que permite a los servidores enviar actualizaciones al cliente mediante una conexión HTTP unidireccional. A través de esta técnica un servidor puede enviar actualizaciones en tiempo real a un cliente, sin que este tenga que solicitarlas continuamente. Los datos son enviados en formato de eventos, es decir se envían pequeños paquetes en lugar de un gran bloque de datos, lo que permite al cliente procesar los datos de manera eficiente. SignalR utiliza esta tecnología cuando los WebSockets no están disponibles, ya que es más eficiente que el Long Polling.

Forever Frame: Técnica de comunicación en tiempo real que solo es compatible con Internet Explorer, y utiliza un IFrame oculto y una conexión unidireccional para transmitir datos desde el servidor al cliente. Sin embargo, debido a su incompatibilidad con otros navegadores y a su ineficiencia en términos de uso de recursos del servidor, ha sido reemplazado por otras tecnologías más modernas y eficientes.

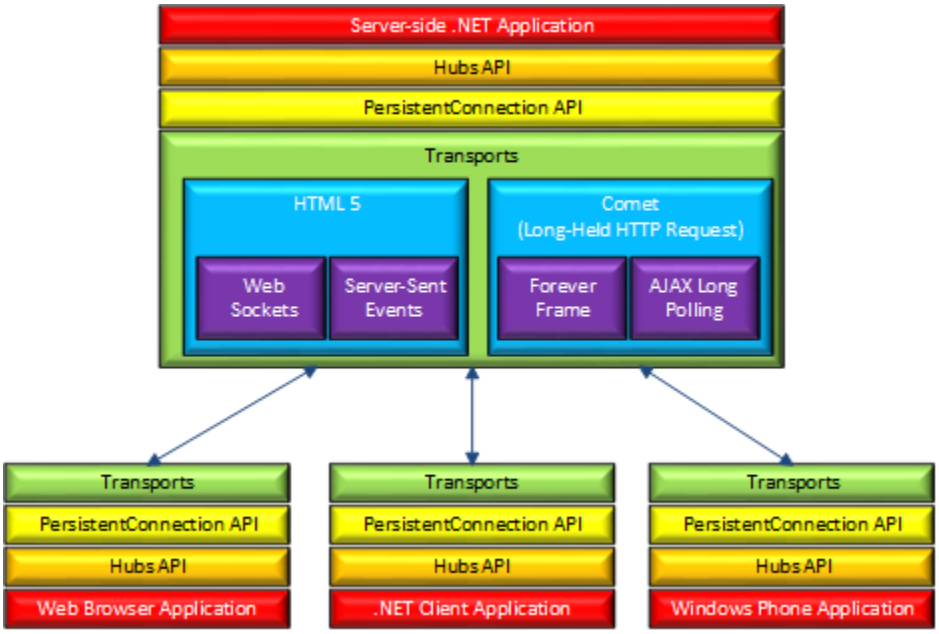
Long Polling: Esta técnica utiliza una conexión HTTP que se mantiene abierta durante un tiempo determinado para recibir actualizaciones en tiempo real. SignalR utiliza Long Polling cuando ni WebSockets ni SSE están disponibles en el cliente o en el servidor.

La API de SignalR contiene dos modelos para la comunicación entre clientes y servidores: conexiones persistentes y concentradores.

Concentradores

Un concentrador es una canalización de más alto nivel basada en la API de conexión que permite al cliente y al servidor llamar métodos entre sí directamente. SignalR maneja el envío a través de los límites de la máquina como por arte de magia, lo que permite a los clientes llamar a métodos en el servidor tan fácilmente como métodos locales, y viceversa.

El uso de un concentrador también le permite pasar parámetros fuertemente tipados a los métodos, lo que permite el enlace de modelos.



El funcionamiento de los hubs, se describe como el código del lado del servidor que llama a un método en el cliente, envía un paquete a través del transporte activo que contiene el nombre y los parámetros del método al que se va a llamar (cuando un objeto se envía como parámetro de método, se serializa mediante JSON). A continuación, el cliente hace coincidir el nombre del método con los métodos definidos en el código del lado cliente. Si hay una coincidencia, el método de cliente se ejecutará utilizando los datos del parámetro deserializado.

La llamada al método se puede monitorear usando herramientas como Fiddler. La siguiente imagen muestra una llamada a método enviada desde un servidor SignalR a un cliente de explorador web en el panel Registros de Fiddler. La llamada al método se envía desde un concentrador llamado , y el método que se invoca se llama .MoveShapeHubupdateShape

```
10:38:03:1298 Session846.WebSocket'WebSocket #846' - Pushing 104 bytes from server WebSocket
81 66 7B 22 43 22 3A 22 42 2C 31 35 7C 43 2C 30     f{"C":"B,15|C,0
7C 44 2C 30 7C 45 2C 30 22 2C 22 4D 22 3A 5B 7B  |D,0|E,0","M":[{
22 48 22 3A 22 4D 6F 76 65 53 68 61 70 65 48 75  "H":"MoveShapeHu
62 22 2C 22 4D 22 3A 22 75 70 64 61 74 65 53 68  b","M":"updateSh
61 70 65 22 2C 22 41 22 3A 5B 7B 22 6C 65 66 74  ape","A":[{"left
22 3A 35 30 31 2E 30 2C 22 74 6F 70 22 3A 33 30  ":501.0,"top":30
32 2E 30 7D 5D 7D 5D 7D                          2.0}]}]}
```

En este ejemplo, el nombre del concentrador se identifica con el parámetro;

Para utilizar las API de Hubs

- Es necesario especificar el formato del mensaje real enviado.
- Trabajar con un modelo de mensajería y distribución en lugar de un modelo de invocación remota.
- Una aplicación existente que utiliza un modelo de mensajería se está portando para utilizar SignalR.

Plataformas soportadas

Servidor

SignalR es compatible con una variedad de configuraciones de servidor y cliente. Además, cada opción de transporte tiene un conjunto de requisitos propios; si los requisitos del sistema para un transporte no están disponibles, SignalR conmutará por error correctamente a otros transportes

A continuación, describiremos los Sistemas operativos de servidor compatibles. ya que se puede hospedar en los siguientes sistemas operativos de servidor o cliente.

Tenga en cuenta que para que SignalR use WebSockets, se requiere Windows Server 2012, Windows Server 2016 o Windows 8 (WebSocket se puede usar en sitios web de Windows Azure, siempre que la versión de .NET Framework del sitio esté establecida en 4.5 y Web Sockets esté habilitado en la página Configuración del sitio).

- Windows Server 2016
- Windows Server 2012
- Windows Server 2008 r2
- Ventanas 10
- Ventanas 8
- Ventanas 7
- Windows Azure

Cliente

SignalR se puede utilizar en una variedad de navegadores web, pero por lo general, sólo se admiten las dos últimas versiones.
Las aplicaciones que usan SignalR en navegadores deben usar jQuery versión 1.6.4 o versiones posteriores principales (como 1.7.2, 1.8.2 o 1.9.1).

SignalR se puede utilizar en los siguientes navegadores:

Microsoft Internet Explorer versiones 11. Solo Windows.
Microsoft Edge (Chromium). Se admiten las versiones de escritorio y móvil.
Mozilla Firefox: versión actual - 1, versiones de Windows y Mac.
Google Chrome: versión actual - 1, versiones de Windows y Mac.
Safari: versión actual - 1, versiones Mac e iOS.
Opera: versión actual - 1, solo Windows.
Navegador Android

Además de requerir ciertos navegadores, los diversos transportes que utiliza SignalR tienen sus propios requisitos. Los siguientes transportes se admiten en las siguientes configuraciones:

Requisitos de transporte del explorador web

Transporte	Internet Explorer	Chrome (Windows o iOS)	Firefox	Safari (OSX o iOS)	Androide
WebSockets	10+	actual - 1	actual - 1	actual - 1	N/A
Eventos enviados por el servidor	N/A	actual - 1	actual - 1	actual - 1	N/A
ForeverFrame	8+	N/A	N/A	N/A	4.1
Sondeo largo	8+	actual - 1	actual - 1	actual - 1	4.1

*: 6+ requerido para una funcionalidad completa.

Azure SignalR Service

Si empezamos a tener un tráfico excesivo de mensajes a nuestro hub ,podría generar un costo alto y tendríamos que ver de buscar un servicio escalable, de ahí surgio Azure SignalR Service . Un servicio 100% compatible con la librería signalR.
Esto simplificaría el proceso de agregar funcionalidad web en tiempo real a las aplicaciones a través de HTTP. Esta funcionalidad en tiempo real permite al servicio enviar actualizaciones de contenido a los clientes conectados, como una aplicación web o móvil de una sola página. Como resultado, los clientes se actualizan sin necesidad de sondear el servidor o enviar nuevas **solicitudes** HTTP de actualizaciones.

Estos son algunos ejemplos en los que puede usar Azure SignalR Service:

Actualizaciones de datos de alta frecuencia: juegos, votaciones, encuestas y subastas.
Paneles y monitoreo: paneles de la empresa, datos del mercado financiero, actualizaciones instantáneas de ventas, tablas de clasificación de juegos multijugador y monitoreo de IoT.
Chat: salas de chat en vivo, bots de chat, atención al cliente en línea, asistentes de compras en tiempo real, mensajeros y chats en el juego.
Ubicación en tiempo real en el mapa: seguimiento logístico, seguimiento del estado de la entrega, actualizaciones del estado del transporte y aplicaciones GPS.
Anuncios dirigidos en tiempo real: anuncios push y ofertas personalizados en tiempo real, y anuncios interactivos.
Aplicaciones colaborativas: coautoría, aplicaciones de pizarra y software de reuniones de equipo.
Notificaciones push: redes sociales, correo electrónico, juegos y alertas de viaje.
Transmisión en tiempo real: transmisión de audio / video en vivo, subtítulos en vivo, traducción y transmisión de eventos y noticias.
IoT y dispositivos conectados: métricas de IoT en tiempo real, control remoto, estado en tiempo real y seguimiento de ubicación.
Automatización: disparadores en tiempo real de eventos ascendentes.

Conclusiones

Si queremos tener soluciones en tiempo real, que aligeren la carga de los backend en los sitios web, con un trabajo más asíncrono desde una solución cliente, podemos utilizar SignalR o Azure SignalR Services.

Referencias

Microsoft: Introduction to SignalR | Microsoft Learn
NetMentor: Aplicaciones en tiempo real con SignalR (netmentor.es)